Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

David Starič

Lokalni razpoznavalnik govora za Android

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Robert Rozman

Ljubljana, 2017

To delo je ponujeno pod licenco Creative Commons Priznanje avtorstva-Deljenje pod enakimi pogoji 2.5 Slovenija (ali novejšo različico). To pomeni, da se tako besedilo, slike, grafi in druge sestavine dela kot tudi rezultati diplomskega dela lahko prosto distribuirajo, reproducirajo, uporabljajo, priobčujejo javnosti in predelujejo, pod pogojem, da se jasno in vidno navede avtorja in naslov tega dela in da se v primeru spremembe, preoblikovanja ali uporabe tega dela v svojem delu, lahko distribuira predelava le pod licenco, ki je enaka tej. Podrobnosti licence so dostopne na spletni strani creativecommons.si ali na Inštitutu za intelektualno lastnino, Streliška 1, 1000 Ljubljana. Izvorna koda aplikacije je objavljena pod licenco GNU GPL in je na voljo na spletnem portalu GitHub [1].

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:
Lokalni razpoznavalnik govora za Android (Local Speech Recognizer for Android)

Tematika naloge:

Izdelajte mobilno aplikacijo za lokalno razpoznavanje govora; proces naj se izvaja brez aktivne spletne povezave in potrebe po velikem številu učnih primerov. Ob tem upoštevajte tudi izobraževalni vidik; uporabnik naj ima možnost vpogleda v delovanje uporabljene metode razpoznavanja in posledično lažje razumevanje celotnega procesa. Delovanje aplikacije poskusite tudi optimizirati, da bo čimbolj uspešna in učinkovita. Preizkusite jo v praksi in analizirajte njeno delovanje.

Zahvaljujem se mentorju viš. pred. dr. Robertu Rozmanu za pomoč, podporo in nasvete pri izdelavi diplomske naloge. Zahvaljujem se tudi staršema in Sonji, za vso pomoč, razumevanje in spodbudo v času študija.

Kazalo

Slike

Tabele

Seznam uporabljenih kratic

kratica	angleško	slovensko
FFT	Fast Fourier transform	Fourierov transform
DTW	Dynamic time warping	dinamično časovno izkrivljanje
FBANK	Filter bank	značilke FBANK
MFCC	Mel-frequency cepstral coefficients	značilke MFCC
DCT	Discrete cosine transform	diskretni kosinusni transform
HMM	Hidden Markov model	skriti Markov model
VCS	Version control software	sistem za verzioniranje kode

Povzetek

Naslov: Lokalni razpoznavalnik govora za Android

Kljub velikem tehnološkem napredku, smo za učinkovito uporabo računalnikov še vedno primorani uporabljati enostavne načine vnašanja ukazov. Ko pa se prestavimo v svet mobilnih naprav nivo interakcije sunkovito zraste. Veliko uporabnikov mobilnih naprav bi želelo uporabljati zvočne ukaze tako za preprosta (ukaz za klicanje), kot tudi za zahtevnejša opravila, kot je spletno iskanje. Za razpoznavanje in izvajanje takih ukazov pa je potrebno veliko število učnih primerov na katerih se razpoznavalnik uči in izboljšuje. Temu dejstvu se z diplomsko nalogo skušamo izogniti. Tako je bil cilj diplomske naloge izdelava mobilne aplikacije, ki zna razpoznati govor brez večje množice učnih primerov in bo delovala povsem samostojno, brez spletne povezave. Za izdelavo razpoznavalnika je bila uporabljena metoda časovnega izkrivljanja. Opravljena pa je bila tudi analiza delovanja, kje smo primerjali točnost in hitrost razpoznavanja. Na podlagi rezultatov analize smo nato z optimizacijo razpoznavanja pohitrili delovanje razpoznavalnika in ob tem ohranili njegovo uspešnost.

Ključne besede: DTW, razpoznavalnik govora, parametrizacija, MFCC.

Abstract

Title: Local speech recognition for Android

Despite big technological advancements, we are still bound to use keyboard and mouse as default computer peripheral. When using mobile devices, the scope of possible interactions with a device quickly grows. Many mobile device users would like to use voice commands for both simple (voice dial), as well as for demanding tasks such as web search. Usually, it takes a number of learning examples on which speech recognizer learns and improves. In this thesis, we are trying to avoid this necessity. The main goal of this thesis is the development of a mobile application that recognizes speech without a demand for a substantial number of learning examples and which will operate completely independently, without active internet connection. The method used for speech recognition system in this thesis is well-known Dynamic Time Warping method. Analysis, where we compared accuracy and speed of recognition, was carried out. Based on the results of the analysis we have successfully proposed a few steps to optimize the speed of our speech recognition application and preserve its accuracy.

Keywords: DTW, speech recognition, speech parameterization, MFCC.

Poglavje 1

Uvod

Z hitrim porastom deleža pametnih mobilnih naprav in njihovo zmogljivostjo, se vse več problemov, ki so bili do sedaj v domeni računalnikov, rešuje prav z njimi. Tako lahko z eno samo napravo predvajamo glasbo v ozadju, pošiljamo elektronsko pošto in brskamo po spletu. Vseeno pa so nekatere stvari preveč odvisne od internetne povezave. Med te spada tudi razpoznavanje govora, ki z omogočeno internetno povezavo, na tak ali drugačen način lajša interakcijo osebe z mobilno napravo.

Tako sta vodilni aplikaciji med razpoznavalniki Siri na mobilni platformi iOS ter Google Now na platformi Android. Oba za boljšo uporabo in delovanje potrebujeta internetno povezavo, v nekaterih primerih pa delujeta tudi brez nje. Taki razpoznavalniki delujejo na principu zajema zvoka, pošiljanja podatkov preko interneta in razpoznavanje zajetega zvoka na oddaljeni lokaciji s pomočjo velike množice učnih primerov. Zaradi zgoraj omenjenih lastnosti razpoznavalnikov, smo se pri diplomski nalogi odločili za izdelavo lokalnega razpoznavalnika. Ta za delo ne potrebuje ogromne zbirke učnih primerov za razpoznavanje in aktivne internetne povezave. Poleg razpoznavalnika je bil v sklopu aplikacije razvit tudi prikaz spektrogramov, ki uporabniku aplikacije omogoča hitrejše in lažje razumevanja postopka parametrizacije zvočnega signala.

V nadaljevanju besedila se v prvem poglavju srečamo z orodji uporabljenimi za razvoj aplikacije. V drugem in tretjem poglavju se dotaknemo teoretične podlage nujne za razumevanje in razvoj aplikacije. V četrtem poglavju diplomske naloge pa je opisana zgradba aplikacije ter njen razvoj. Pri razvoju aplikacije smo se srečali tudi z problematiko optimizacije delovanja, čemur je namenjeno zadnje poglavje. Na koncu diplomske naloge pa so zapisane sklepne ugotovitve in možne nadgradnje aplikacije.

Poglavje 2

Uporabljena orodja in tehnologije

Pri izdelavi aplikacije smo se zaradi velikega deleža uporabnikov in dostopnosti naprav, odločili za platformo Android. Tako pri izdelavi aplikacije z uporabljeno minimalno verzijo sistema 5.0 pokrijemo 63% delež vseh Android naprav v uporabi. Android OS pa je tudi največkrat uporabljen mobilni operacijski sistem v Sloveniji.

Ob dobro dokumentiranem API-ju ter priročnosti testiranja aplikacije na večjem številu naprav, je bila odločitev za uporabo Android platforme enostavna [2].

Zaradi lažjega sledenja stanja implementacije aplikacije je bila uporabljena razvojna strategija SCRUM, kateri smo sledili s spletno aplikacijo Taiga.io.

2.1 Operacijski sistem Android

Android je mobilni operacijski sistem, ki ga razvija podjetje Google in v osnovi bazira na linux platformi. Njegovo arhitekturo tako predstavlja 5 ključnih delov:

- linux jedro (angl. *kernel*)
- knjižnice
- ogrodje aplikacij
- aplikacije
- prevajalnik

Ob veliki popularnosti operacijskega sistema Android se Googlov razvoj ni ustavil samo pri mobilnih napravah, temveč sedaj obsega tudi pametne televizijske sprejemnike, ure in avtomobile. Če odmislimo mikrokrmilniške operacijske sisteme, se Android OS ponaša z nazivom največkrat nameščenega operacijskega sistema na elektronskih napravah.

2.2 Sistem za upravljanje z izvorno kodo - Git

Je trenutno daleč najbolj uporabljen sistem za verzioniranje kode (VCS). Je odprto kodni sistem, ki se aktivno razvija in ga je leta 2005 razvil Linus Torvalds. Glavna prednost uporabe Git-a v primerjavi z drugimi sistemi verzioniranja je predvsem distribuiran sistem, kar omogoča da zgodovina sprememb ni centralizirana in je dostopna vsakemu uporabniku, ki uporablja isto delovno kopijo kode [3].

2.3 Orodje za vodenje projektov - Taiga.io

Taiga je sistem za vodenje projektov. Omogoča lažje in hitrejše vodenje projektov, enostaven pregled nad aktivnimi opravili ter lažje spremljanje časovne zahtevnosti projekta. Trenutno je za uporabo namenjena samo internetna aplikacija, ki omogoča neomejeno število odprtih projektov, privatni projekti pa so plačljive narave.

Pri izdelavi diplomske naloge smo si s tem sistemom veliko pomagali ter zapisali vse večje module kot aktivnosti, ki ji je potrebno dokončati. Vse aktivnosti smo nato razbili na več manjših aktivnosti in za vsako zapisali koliko časa potrebujemo za realizacijo. Ob upoštevanju metodologije razvoja SCRUM, smo nato aktivnosti razdelili v časovna obdobja (angl. *sprint*) v katerih jih je bilo potrebno zaključiti [4]. Z uporabo te metodologije, smo vedno imeli dober pregled nad aktivnostmi, ki so potrebovale našo pozornost. Z identifikacijo in razbitjem zahtevnejših aktivnosti, ki bi lahko povzročile težave in s tem upočasnile razvoj, pa nam je uspelo pravočasno zaključiti z izdelavo aplikacije.

Poglavje 3

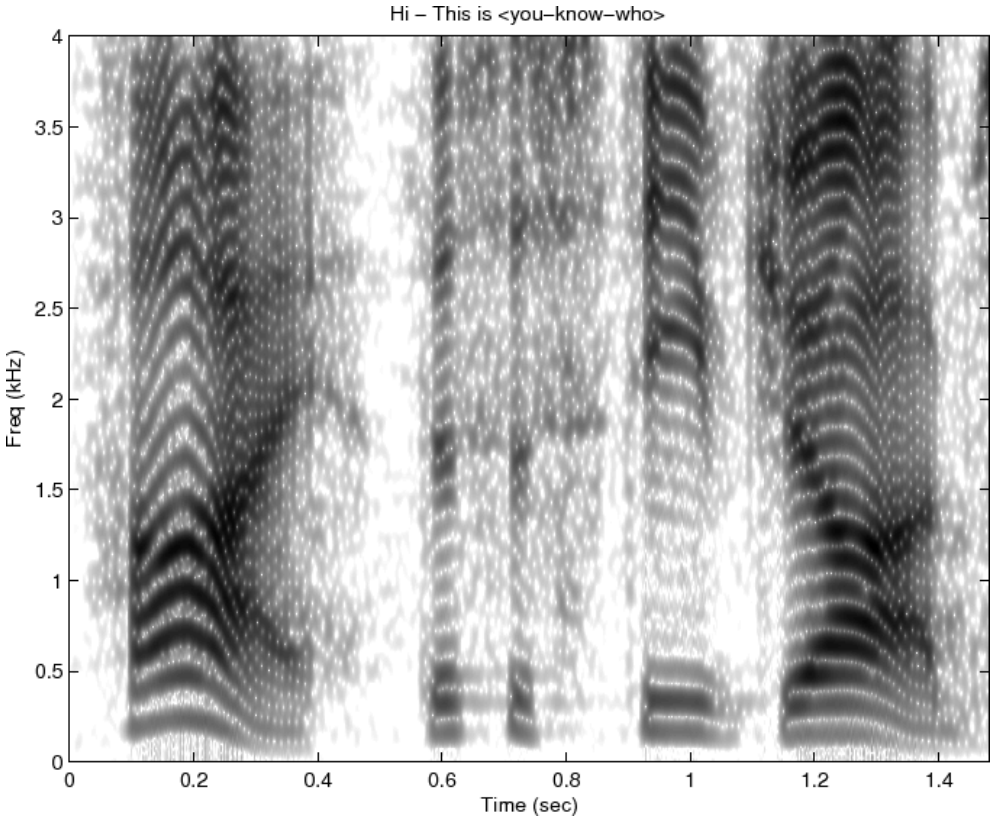
Parametrizacija zvočnih signalov

Pri izdelavi sistema za razpoznavanje govora se srečujemo s problemom pridobivanja relevantih podatkov iz zajetega govornega signala. Zato se bomo v poglavju parametrizacije zvočnih signalov dotaknili teoretičnega dela diplomske naloge. V prvem podpoglavju je prikazan postopek frekvenčne analize z izračunom t.i. kratkočasovnega amplitudnega odziva, v drugem pa sledi izračun značilk. Poglavje zaključimo s sestavo vektorja značilk, ki odraža strnjen zapis govornega signala. Tako parametriziran zapis se potem uporabi na poljubni metodi razpoznavanja.

3.1 Frekvenčna analiza

Ko govorimo o frekvenčni analizi signala največkrat pomislimo na proces, pri katerem z uporabo hitre Fourierove transformacije (v nadaljevanju FFT), izračunamo kratkočasovni amplitudni odziv - spektrogram (slika 3.1). To storimo na naslednji način. Vzorčene podatke signala iz časovne domene najprej razdelimo na več okvirjev, ki se medsebojno prekrivajo. Z uporabo FFT izračunamo vrednosti frekvenčnega odziva posameznega okvirja. Iz frekvenčnih odzivov nato izračunamo absolutne vrednosti, ki predstavljajo

amplitudne odzive okvirjev [5]. Vsak tak okvir je v izrisanem spektrogramu prikazan z eno vertikalno črto, ki prikazuje porazdelitev frekvenc v določenem časovnem trenutku. Vrednosti amplitud so v spektrogramu predstavljene z barvnimi odtenki. Ti spektri so velikokrat uporabljeni za izdelavo tri dimenzionalnega prikaza, ki ga običajno imenujemo spektrogram.



Slika 3.1: Primer spektrograma govornega signala.

3.2 Vektorji značilk

Za osnovno razpoznavanje govora je tako izračunan spektrogram dovolj. V primeru izdelave bolj robustnega razpoznavalnika pa spektrogram ni najbolj primeren, ker vsebuje preveč individualnih govornih značilnosti govorca. Zato je potrebno informacije, ki jih pridobimo s spektrograma posplošiti,

da zajemajo manj individualnih lastnosti govorca in več splošnih lastnosti. To ponavadi pomeni združevanje v frekvenčne pasove in izračun posebnih značilk, ki bolj robustno opisujejo lastnosti govora [6].

3.2.1 Značilke FBANK

Spektrogram vsebuje preveč odvečnih informacij za razpoznavanje govora, predvsem zato, ker uho ne zazna razlike med dvema bližnjima frekvencama, ta efekt pa se samo še stopnjuje pri višjih frekvencah. Zato združujemo energije okvirja v frekvenčne pasove, da dobimo boljšo predstavo koliko energije je v posameznih pasovih. To nam omogoča uporaba melodične ("Mel") frekvenčne lestvice in z njo povezanih frekvenčnih pasov. Na primer: prvi frekvenčni pas nam pove koliko energije obstaja blizu frekvence 0Hz. Ko se frekvence višajo se tudi širina frekvenčnega pasu viša, ker človekov sluh enostavno ne razlikuje tako dobro višjih frekvenc. Mel lestvica nam pove kako moramo razdeliti frekvenčne pasove, da s tem čim bolje posnemamo percepcijo človeškega ušesa.

Z uporabo enačbe za izračun porazdelitve frekvenčnih pasov dobimo trikotne filtre, ki se medsebojno prekrivajo in tako nelinearno pokrijejo področje človeškega sluha. Tipično je število frekvenčnih pasov od 19 do 24. (slika 3.2)

S pomočjo frekvenčnih pasov lahko nato izračunamo vrednosti značilk FBANK. Za izračun potrebujemo amplitudne odzive okvirjev, na katerih uporabimo izračunane frekvenčne pasove. Tako nam za vsak okvir ostane 19 do 24 vrednosti, ki predstavljajo energijo v tistem frekvenčnem pasu. Pri končnem izračunu značilk FBANK moramo tudi upoštevati da se izhodi filtrov prekrivajo, in s tem vsak izhod filtra poda vsebino kritičnega pasu kot vsoto sosednjih izhodov filtrov.

Tako pridobljene vrednosti značilk FBANK je potrebno logaritmirati. Loga-

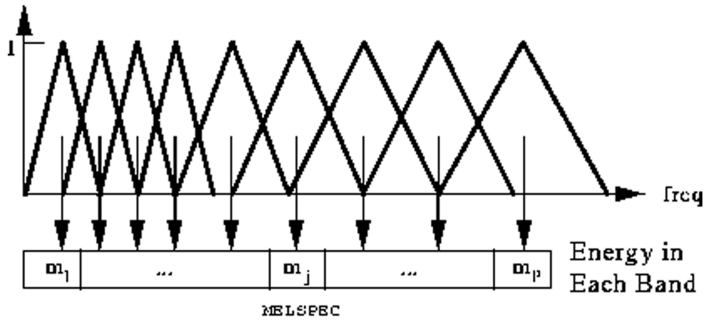


Fig. 5.3 Mel-Scale Filter Bank

Slika 3.2: Prikaz delitve signala v več nelinearnih frekvenčnih pasov [12].

ritmiranje vrednosti značilk je ravno tako povezano z človeških sluhom, saj glasnost zvoka ne slišimo linearno. To pomeni, da večje spremembe energije značilk ne predstavljajo tako velike spremembe, če je začetni zvok prav tako glasen. Tako se z logaritmiranjem naše značilke mnogo bolje prilegajo človeškemu zaznavanju zvoka.

3.2.2 Značilke MFCC

Problem značilk FBANK predstavlja predvsem korelacija med njimi, ker se frekvenčni pasovi, ki so uporabljeni za izračun, medsebojno prekrivajo. Uporaba diskretnega kosinusnega transforma, v nadaljevanju DCT, na značilkah FBANK dekorelira vrednosti značilk, kar pomeni da za razpoznavanje lahko uporabimo statistične razpoznavalnike, ki temeljijo na skritem Markovem modelu (HMM). Z uporabo DCT prav tako pridobimo na koristni variabilnosti. Tako nižji koeficienti vektorja značilk vsebujejo značilnosti govornega trakta, kar je ključ do razpoznavanja vsebine govora [9]. Ker pa se izkaže da je zadnji del vektorja značilk MFCC sestavljen predvsem iz hitrih sprememb energije značilk FBANK, ki znižajo zmogljivost razpoznavalnika, ta del vektorja značilk zavržemo. Tako dobimo nov vektor značilk MFCC dolžine 12, ki mu ponavadi na začetku pripnemo še energijo okvirja. S tem, ko smo ohranili

zgolj prvi 12 vrednosti, smo tudi nekoliko zmanjšali velikost vektorja značilk (iz prvih 19-24), ter s tem "zgladili" spekter signala [10].

3.2.3 Dinamične značilke MFCC delta in delta+delta

Predstavljajo spremembe in časovni potek v značilkah. Značilke MFCC predstavljajo energijo enega okvirja, vendar govor ne predstavljajo samo okvirji temveč tudi dinamika med njimi. Zato izračunamo značilke delta, ki predstavljajo projekcije med okvirji značilk, te pa vsebujejo informacijo o hitrosti govora. Izkaže se, da izračunane značilke delta poleg značilk MFCC ogromno pripomorejo k zmogljivosti razpoznavanja govora. Zaradi izboljšanja razpoznavanja jih dodamo v naš vektor značilk okvirja, ki sedaj z značilkami delta šteje 24 vrednosti. Za izračun značilk delta se uporablja naslednji izraz:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}, \tag{3.1}$$

kjer d_t predstavlja delta koeficient, N je tipično 2, c_{t+n} in c_{t-n} pa predstavljata vrednosti levo in desno od trenutne značilke MFCC.

Poleg značilk delta se izračuna tudi značilke delta+delta, ki predstavljajo potek trenda značilk MFCC njihove vrednosti pa lahko interpretiramo kot pospeševanje govora. Izračun je enak kot pri izračunu značilk delta, le da namesto značilk MFCC uporabimo značilke delta.

Po zaključeni parametrizaciji nam tako ostane vektor značilk sestavljen iz:

- 12 značilk MFCC
- 1 energija okvija
- 12 značilk delta MFCC
- 1 energija delta okvirja

- 12 značilk delta+delta MFCC
- 1 energija delta+delta okvirja

Skupno 39 značilk na okvir.

Z izračunom značilk MFCC, delta MFCC in delta+delta MFCC smo tako zaključili s postopkom parametrizacije. Tako strnjen zvočni zapis je mnogo bolj splošen kot spektrogram in se lahko uporabi pri različnih metodah razpoznavanja.

Poglavje 4

Razpoznavanje govorjenih besed

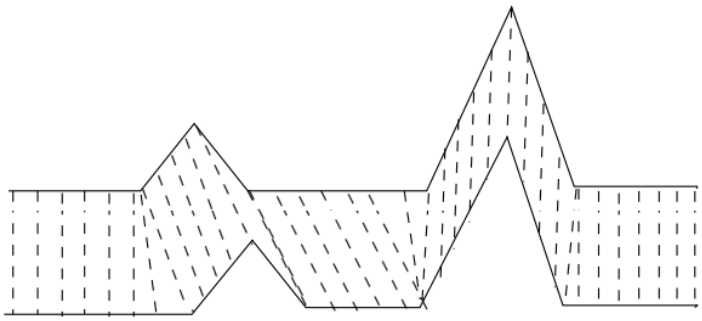
Poznamo več pristopov k razpoznavanju govora. Med najbolj uporabljene spadata algoritem časovnega izkrivljanja (DTW) in statistični modeli (HMM). Prav tako pa med uporabne metode za razpoznavanja zvočnih signalov spadata nevronske (MLP) in globoke mreže (DBN) [11].

Metode razpoznavanja niso omejene na razpoznavanje govora in so splošno uporabne.

4.1 Metoda časovnega izkrivljanja - DTW

Algoritem časovnega izkrivljanja ali *dynamic time warping* algoritem (v nadaljevanju DTW), je algoritem ki bazira na dinamičnemu programiranju. S pomočjo katerega izračunamo podobnost med dvema zaporedjema, ki sta lahko različno dolga. Glavna prednost DTW pred drugimi metodami razpoznavanja govora pa je predvsem ta, da za delovanje ne potrebuje velike učne množice. Zaradi tega in drugih značilnosti se DTW ne uporablja zgolj pri analizi zvočnih signalov temveč tudi pri videu in drugih linearnih zaporedjih [7]. Najbolj uporabljena pa je pri avtomatski razpoznavi govora, kjer

te lastnosti s pridom izkoriščamo pri analizi zvoka počasnejših in hitrejših govorcev (slika 4.1).



Slika 4.1: Prikaz raztezanja in krčenja dveh signalov [7].

Algoritem časovnega izkrivljanja je zgrajen okoli metode, ki išče časovno poravnano pot z največjim možnim ujemanjem med dvema danima zaporedjema. Pri tem dele vhodnih zaporedij raztezamo in krčimo, s ciljem da pridobimo čim boljše ujemanje.

4.2 Matrika podobnosti

Uporaba algoritma časovnega izkrivljanja vključuje tudi izračun matrike podobnosti. Za zaporedji, ki ju preverjamo, je potrebno izračunati razdalje med vektorji. Na ta način pridobimo matriko podobnosti, ki jo potrebujemo za nadaljnjo prepoznavo zvočnega signala. Za izračun razdalje oziroma podobnosti med vektorjema imamo na voljo več različnih metod. Spodaj sta predstavljeni dve najbolj uporabljeni.

- Kosinusna razdalja: Je mera podobnosti med dvema vektorjema, bolj natančno notranjega produkta, ki meri kosinus kota med dvema vektorjema. Kosinus 0° je enak 1 in manj kot 1 pri katerem koli drugem

kotu. Torej imata dva vektorja, ki sta podobno usmerjena kosinusno razdaljo »podobnost« bližjo 1, pravokotna vektorja pa podobnost 0.

- Evklidska razdalja: Je razdalja v evklidskem prostoru, ki nam pove kako oddaljena sta dva vektorja en od drugega.

Izbrana metoda izračuna razdalje se uporabi pri izračunu optimalne poti po matriki podobnosti. Tako lahko za vse sosednje elemente izračunamo razdaljo, in se na podlagi tega odločimo v katero smer bomo nadaljevali z izračunom optimalne poti.

4.3 Dinamično programiranje

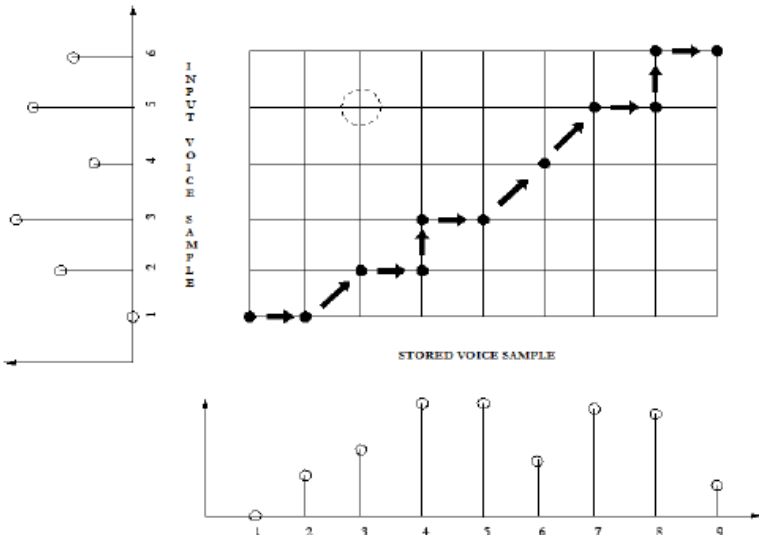
Časovno poravnavo dveh zaporedij rešujemo z dinamičnim programiranjem. Vzamemo izračunano matriko podobnosti, ki tako na svojih oseh predstavlja zaporedja, katerih podobnost nas zanima. Mesto v matriki M(x,y) pa tako predstavlja podobnost vektorja za dan x in y. S postopkom dinamičnega programiranja nato poizkušamo pridobiti optimalno oceno podobnosti med podanima zaporedjema.

```
int DTWDistance(s: array [1..n], t: array [1..m]) {
  DTW := array [0..n, 0..m]

  for i := 1 to n
      DTW[i, 0] := infinity
  for i := 1 to m
      DTW[0, i] := infinity
  DTW[0, 0] := 0

  for i := 1 to n
      for j := 1 to m
          cost := d(s[i], t[j])
          DTW[i, j] := cost + minimum(DTW[i-1, j ], // insertion
```

```
                                DTW[i  , j-1],   // deletion
                                DTW[i-1, j-1])   // match

   return DTW[n, m]
}
```

Za lažjo predstavo izračuna optimalne poti uporabimo spodnji primer. Da dobimo oceno podobnosti moramo zgraditi pot skozi matriko podobnosti. Pot naj se začne pri M(0,0) kjer je M matrika podobnosti, želimo pa doseči cilj M(m,n) kjer sta m in n dolžine posameznih signalov (slika 4.2). Pred tem pa je potrebno postaviti nekaj omejitev za pot po kateri želimo iti.

- pot se začne pri M(0,0) in konča pri M(m,n)
- pot vedno teče naprej, kar pomeni da se iz točke M(i,j) lahko premaknemo desno M(i+1,j), navzgor M(i,j+1) ali diagonalno M(i+1,j+1)



Slika 4.2: Iskanje optimalne poti med dvema signaloma [8].

Te omejitve preprečujejo, da se vrtimo v krogu in problem dinamičnega programiranja prestavi v zgornjo mejo časovne zahtevnosti O(mn).

Pri iskanju optimalne poti uporabljamo strategijo vračanja (angl. *back-tracking*), ki je sorazmerno enostavna in vključuje premik nazaj iz zadnje točke M(m,n) ter iskanje "najcenejše"poti, ki jo lahko dosežemo. Postopek ponavljamo dokler ne dosežemo začetka M(0,0).

Pohitritev izračuna lahko izvedemo tako, da omejujemo iskalni prostor saj s tem zmanjšujemo računsko zahtevnost. Vendar lahko z omejevanjem iskalnega prostora izločimo najboljšo, najcenejšo pot, ker ne iščemo širše od omejenega prostora.

Rezultat izračuna optimalne poti nad podano matriko podobnosti z dinamičnim programiranjem nam vrne oceno podobnosti med dvema zaporedjema.

Tako razpoznavalnik največkrat uporabljamo za prepoznavanje podobnosti zajetega zvoka z zvoki, ki so že nekje shranjeni. V takem primeru ponovimo parametrizacijo za vsak zvok, zgradimo matriko podobnosti s parametriziranim zajetim zvokom in parametriziranim shranjenim zvokom, izračunamo optimalno pot in shranimo rezultate. Nato nam preostane samo še to, da poiščemo pot najboljšega ujemanja in s tem pridobimo podatek o tem, kateri zvok najbolj ustreza zajetemu.

Poglavje 5

Razvoj mobilne aplikacije

V tem delu diplomske naloge bo predstavljeno načrtovanje glavnih modulov, ki so uporabljeni v aplikaciji. Zaradi lažje izdelave in posledično združevanje je razvoj aplikacije razdeljen na:

- raziskava primernih formatov za zapis zajetega zvoka ter izdelava funkcionalnosti, ki jih potrebujemo za delo z izbranim formatom (branje, pisanje podakov)
- modul za zajem podatkov
- razpoznavalnik (parametrizacija zajetih signalov, izračun algoritma časovnega izkrivljanja)
- prikaz rezultatov
- shranjevanje parametriziranih vrednosti v predpomnilnik
- optimizacija razpoznavalnika

V nadaljevanju poglavja so vsi našteti moduli podrobneje opisani.

5.1 Format zapisa ter vhodno izhodne funkcionalnosti

Glavna naloge aplikacije je zajem in razpoznavanje govora. Zajeti govor je potrebno zapisati, pri tem pa je potrebno paziti predvsem na pravilen format zapisa. Poznamo nekompresirane formate, kompresirane vendar brezizgubne formate ter formate z izgubno kompresijo. Glede na potrebe aplikacije se vsi zajeti zvoki shranjujejo v brezizgubnem formatu .wav, kar pomeni, da je bilo potrebno pripraviti modul, ki zna delati s tem formatom. Za ta namen aplikacija uporablja javanski razred WaveTools. Razred omogoča branje .wav datotek na podlagi imen, vrne pa vsebino datoteke v obliki polja vrednosti tipa float. Prej omenjeni razred omogoča tudi zapis .wav datotek v pomnilnik z dodajanjem informacij, značilnih za format wav, v glavo datoteke.

5.2 Modul za zajem podatkov

Aplikacija omogoča zajem zvočnega signala preko vgrajenega mikrofona naprave. Za uspešno uporabo je tako potrebno omogočiti zajem zvoka z uporabo pravice *android.permission.RECORD_AUDIO*. Pred začetkom zajema zvoka je potrebno pravilno inicializirati android snemalnik. Tako so za zajem zvoka uporabljeni sledeči parametri:

- vzorčenje s frekvenco 8000Hz, nudi dovolj dobro razmerje med kvaliteto in hitrostjo procesiranja. Povprečne frekvence govora pa se gibljejo do 4000Hz kar po Nyquistovem teoremu privede k frekvenci vzorčenja 8000Hz
- 16-bitno PCM kodiranje, kar pomeni da iz mikrofona zajeti vzorec shrani kot 16bitno predznačeno celo število. Vsak vzorec lahko tako zavzame vrednost med -32768 in 32767.
- mono vhodni kanal, za zajem zvoka se uporabi en sam kanal. Večina mobilnih telefonov ne omogoča stereo zajema zvoka preko vgrajenega

mikrofona.

Na tem mestu je vse pripravljeno za zajem signala, potrebujemo le še mesto za hranjenje naših posnetkov. Aplikacija za pravilno delovanje potrebuje dovoljenje za pisanje v zunanji pomnilnik naprave, torej v pomnilnik, ki se ne briše z izbrisom aplikacije. Z omogočenim dovoljenjem android.permission.WRITE_EXTERNAL_STORAGE, aplikacija na zunanjem pomnilniku ustvari mapo AudioRecorder, kamor se shranjujejo posnetki.

Aplikacija nato ob kliku na gumb za zajem zvoka ustvari novo procesno nit. Nato se iz snemalnega objekta, glede na količino medpomnilnika, ki ga Android naprava premore, v zanki črpa podatke iz mikrofona. Ob ponovnem pritisku na snemalni gumb aplikacija vpraša uporabnika za ime datoteke in zapiše zajeti zvok. Podatke se nato prepiše v datoteko formata .wav; ob tem pa se v glavo datoteke pripne še zaporedje podatkov, ki vključuje skupno dolžino teh podatkov, število uporabljenih kanalov, frekvenco vzorčenja, kodiranje in skupna dolžina audio podatkov. Vse skupaj se nato shrani v zunanji pomnilnik naprave, kjer čaka na nadaljnjo uporabo.

5.3 Modul razpoznavalnika

Cilj diplomske naloge je bila izdelava razpoznavalnika govora, ki za svoje delovanje ne potrebuje aktivne povezave z internetom in omogoča dovolj robustno prepoznavo govornih signalov. Prav tako mora delovati brez množice učnih primerov, kot to počnejo vsi večji tekmeci na trgu razpoznavalnikov. Modul razpoznavalnika se tako deli na več manjših delov:

- parametrizacija vhodnih podatkov
- preračunavanje matrik podobnosti
- izračun podobnosti med dvema signaloma

Za potrebe tega modula je bil razvit razred calcSpec, ki omogoča obravnavo vseh zgoraj naštetih delov razpoznavalnika. Zaradi večje časovne zahtevnosti

razred razpoznavalnika razširja abstrakten razred AsyncTask(Params, Progress, Result). Tako izpeljan razred lahko opravlja svoje delo v ozadju, brez da bi s tem prekinjal glavno procesno nit aplikacije, kar pripomore k boljši uporabniški izkušnji. Ob uspešno zaključenem delu aplikacija uporabnika obvesti preko obvestilne vrstice na spodnjem delu zaslona.

5.3.1 Parametrizacija vhodnih podatkov

Glavni cilj parametrizacije je pridobiti čim bolj koristne informacije iz shranjenega zvočnega zapisa. Tako aplikacija pri parametrizaciji uporablja knjižnico za izračun hitre Fourierove transformacije. Knjižnica omogoča izvajanje Fourierove transformacije brez dodatne alokacije pomnilnika.

Aplikacija tako pri frekvenčni analizi z FFT uporablja naslednje parametre. Frekvenca vzorčenja, ki je v primeru aplikacije privzeto nastavljena na 8000Hz, kar predstavlja dobro razmerje med hitrostjo in kvaliteto. Za število vhodnih vzorcev ta implementacija FFT potrebuje 2^n vzorcev, parameter aplikacije pa je nastavljen na 512.

Da lahko analiziramo zvočni signal dovolj natančno ga je potrebno razbiti na več manjših okvirjev, ki se medsebojno prekrivajo. Pri govoru se tako uporablja kot parameter velikosti okvirja 20-40ms. Prav tako je potrebno upoštevati zamik okvirja, ki je v aplikaciji definiran na 4ms, kar z uporabljeno velikostjo okvirja 32ms predstavlja 87.5% prekrivanje okvirja.

Ker pa vzorčenje zvočnega signala prinese tudi spektralno odtekanje se za zmanjšanje le tega čez okvirje preračuna še Hammingovo okno.

```
public float[] hamming(int len){
        float [] win = new float [len];
        for (int i = 0; i<len; i++){
            win[i] = (float)
                (0.54-0.46*Math.cos((2*Math.PI*i)/(len-1)));
        }
        return win;
    }
```

Aplikacija nato za podatke preko FFT izračuna transformacijo vsakega okvirja in nato na podlagi rezultatov izračuna absolutne vrednosti okvirjev. Tako pripravljene vrednosti uporabimo za izris kratkočasovnega amplitudnega odziva oz. spektrograma.

Ker je izračunan spektrogram premalo robusten za uporabo v razpoznavalniku, aplikacija nato iz spektrograma izračuna značilke MFCC. Te veliko bolje povzamejo splošne značilnosti govora in s tem izboljšajo delovanje razpoznavalnika.

Prvi del izračuna značilk MFCC je določitev vrednosti značilk FBANK. Za ta namen je bila razvita funkcija, ki pretvori uporabljeno frekvenco vzorčenja in izračuna frekvenčne filtre po Mel lestvici. Filtre nato uporabimo na amplitudnem odzivu okvirja, kar vrne značilke FBANK. Za vsak okvir tako aplikacija preračuna 22 značilk FBANK.

Za izračun značilk MFCC nato logaritmiramo vrednosti značilk FBANK in izvedemo diskretni kosinusni transform oz. DCT nad vrednostmi značilk FBANK. Tako pridobimo vrednosti značilk MFCC, ki so zaradi DCT nekorelirane, kar se spridoma uporablja pri statističnih razpoznavalnikih zvoka. Od izračunanih značilk MFCC obdržimo samo prvih 12. Aplikacija nato izračuna delta in delta+delta značilke MFCC, ki prikazujejo potek in trend

značilk MFCC.

```
double [][] deltas = new double[mfcc.length][mfcc[0].length];
        for (int i = 0; i < mfcc.length; i++) {
            deltas[i] = delta(mfcc[i], 2);
        }
double [][] deltasdeltas = new
    double[deltas.length][deltas[0].length];
   for (int i = 0; i < deltas.length; i++) {
            deltasdeltas[i] = delta(deltas[i], 2);
        }

public static double[] delta( double[] input, int N) {

        int nbrFrames = input.length+ 2*N;
        double[] result = new double[nbrFrames+2*N];
        for (int i = 0; i < nbrFrames; i++) {
            if (i == 0 || i == 1) {
                result[i] = input[0];
            } else if (i == nbrFrames-1 || i == nbrFrames-2) {
                result[i] = input[input.length-1];
            } else {
                result[i] = input[i-2];
            }
        }

        double denom = 0.0;
        for (int i = 1; i < N+1; i++) {
            denom += 2 * i * i;
        }

        double [] deltas = new double[nbrFrames -2*N];
        for (int i = 0; i < nbrFrames-2*N; i++) {
```

```
            double sum = 0.0;
            for (int j = -1*N ; j < N+1; j++) {
                sum += j * result[N + i + j];
            }
            deltas[i] = sum / denom;
        }
        return deltas;
    }
```

Z izračunom delta spremenljivk aplikacija zaključi s parametrizacijo signala. Pridobljene podatke o MFCC, delta MFCC in delta+delta MFCC spremenljivkah nato uporabimo pri nadaljnjem delu.

5.3.2 Preračunavanje matrik podobnosti

Matrika podobnosti je ključnega pomena pri uporabi razpoznavalnika, ki temelji na algoritmu časovnega izkrivljanja. Aplikacija pri delu uporablja dva pristopa k reševanju problema računanja matrik podobnosti. Če v aplikaciji nastavimo, da ne uporabljamo hitrejšega algoritma se matrika podobnosti predhodno izračuna in nato izračunano matriko dostavimo kot vhodni parameter funkciji iskanja optimalne poti. V primeru uporabe hitrejšega algoritma časovnega izkrivljanja, pa razdaljo med dvema vektorjema značilk računamo spotoma, saj v tem primeru uporabljamo Sakoe-Chiba pas, ki znatno zmanjša površino matrike in s tem število opravljenih računskih operacij. Za izračun matrike podobnosti se uporablja naslednja metoda:

```
public static float distance(double[] frame, double[] frame2)
    {
        double tempSum = 0;
        for (int i = 0; i < frame.length; i++)
            tempSum += Math.pow(Math.abs(frame[i] - frame2[i]), 2);
        return (float)(Math.sqrt(tempSum)/1000);
    }
```

Vhodna parametra predstavljata parametriziran okvir sestavljen iz značilk MFCC, delta MFCC in delta+delta MFCC. Rezultat pa predstavlja razdaljo, ki jo uporabimo v matriki podobnosti.

5.3.3 Izračun podobnosti med dvema vhodnima signaloma

Aplikacija računa podobnost med dvema vhodnima signaloma s pomočjo dinamičnega programiranja, ki omogoči iskanja optimalne poti po matriki podobnosti. Kot vhodni parameter sprejme matriko podobnosti, za rezultat pa vrne oceno podobnosti.

Ko pri uporabi aplikacije shranimo zajet zvok, gre le ta skozi proces parametrizacije. Parametriziran signal nato uporabimo kot prvi vhodni podatek funkcije dinamičnega programiranja. Za drugi vhodni podatek pa iteriramo skozi vse do sedaj shranjene zvoke, ki so že parametrizirani. Za vsak par vhodnih podatkov se nato izračuna ocena podobnosti. Z iteracijo čez vse pare vhodnih podatkov pridobimo polje z izračunanimi cenami poti, ki služi za odločanje o tem kateri signal je najbolj podoben iskanemu.

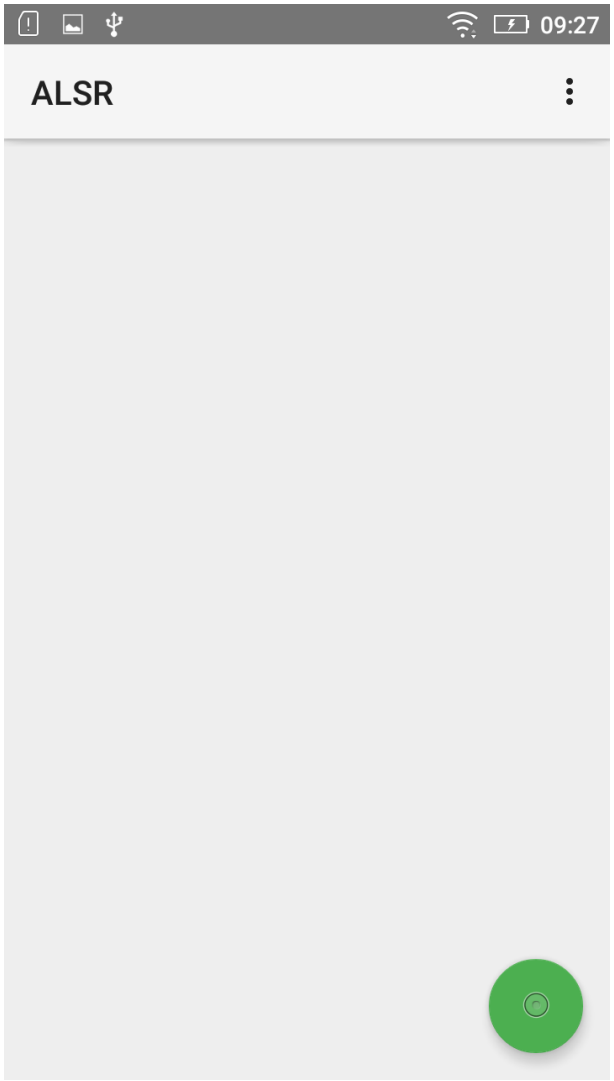
5.4 Prikaz rezultatov

Za aplikacije, ki slonijo na operacijskem sistemu Android, je izgled aplikacije definiran v XML datotekah. Podoba aplikacije je tako razdeljena na tri XML datoteke, ki skrbijo za osnovni zaslon, prikaz rezultatov razpoznavalnika ter prikaz orodne vrstice.

5.4.1 Osnovni zaslon

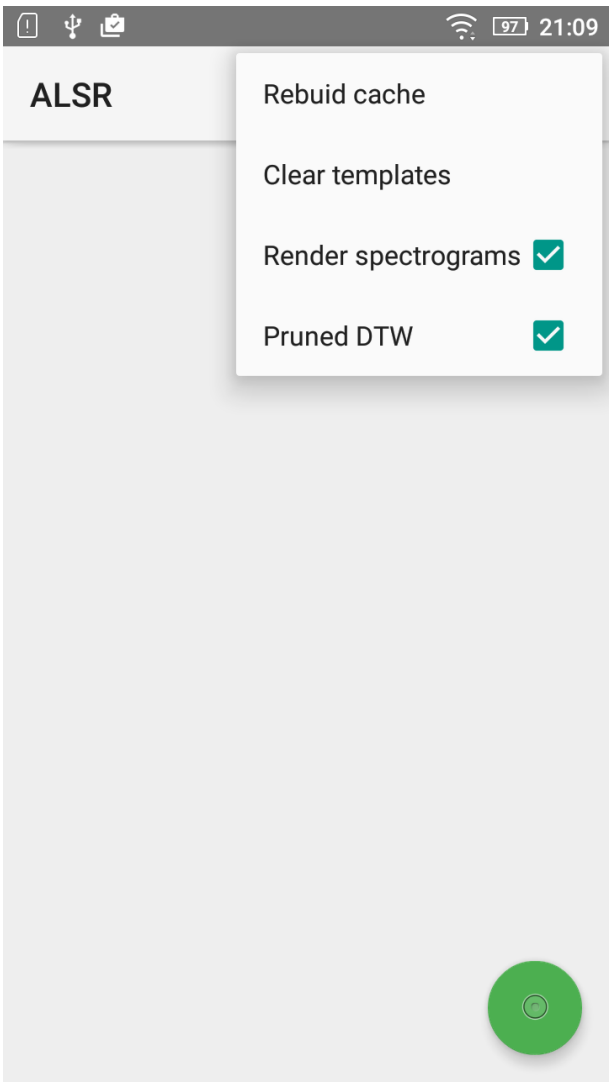
Ko aplikacijo zaženemo, se nam na osnovnem zaslonu prikaže gumb za začetek snemanja novega posnetka ter orodna vrstica. Ob pritisku na gumb za začetek snemanja, uporabnik sproži novo procesno nit, ki začne z zajema-

njem podatkov, dokler uporabnik ponovno ne pritisne na gumb (slika 5.1). Ob tem se odpre novo pojavno okno, ki uporabniku omogoči poimenovanje zajetega posnetka, da ga lahko kasneje identificira.



Slika 5.1: Začetni zaslon aplikacije.

Osnovni gradnik, ki je ravno tako vseskozi prisoten je orodna vrstica. Ta ob pritisku nanjo uporabniku ponudi različne funkcionalnosti in nastavitve aplikacije. Med orodji tako najdemo možnost ponovne izdelave medpomnilnika podatkov ter funkcionalnost brisanja vseh shranjenih podatkov iz medpomnilnika. Prav tako lahko preko orodne vrstice vključimo in izključimo izris spektrogramov in uporabo hitrega razpoznavanja (slika 5.2).



Slika 5.2: Prikaz orodne vrstice ter nastavitev aplikacije.

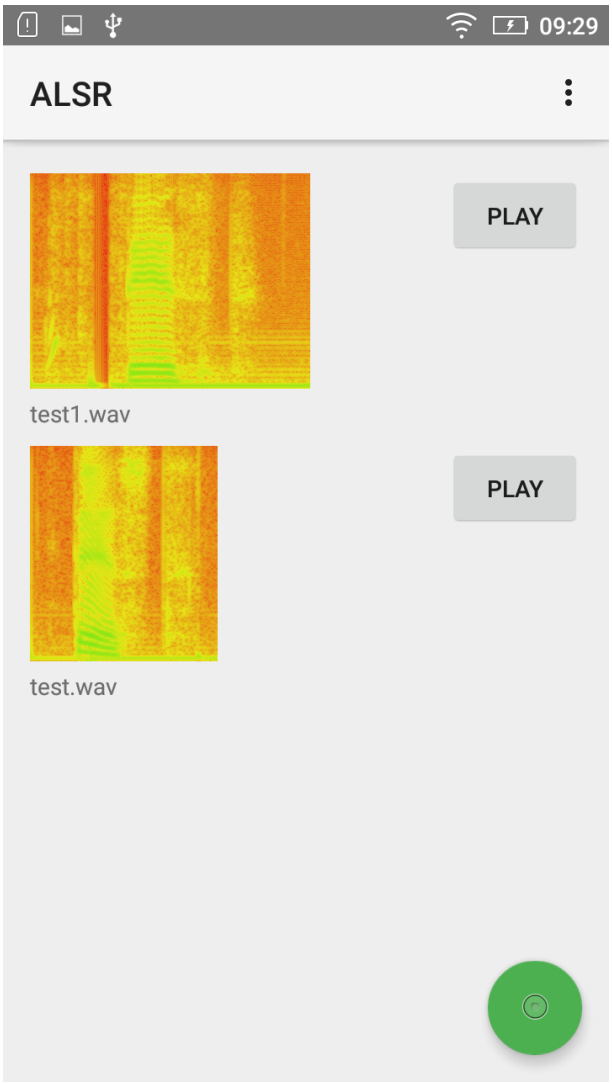
5.4.2 Pregled spektrogramov

Ko uporabnik zaključi z vnosom imena datoteke, aplikacija sporoči uporabniku, da se v ozadju izvaja razpoznavanje. Sporočila, ki jih aplikacija vrača ob delovanju, zaključkih, ali drugih menijskih akcijah uporabljajo razred Snackbar, ki skrbi da so povratne informacije vidne in jih ob neaktivnosti tudi samodejno skrije.

Ko aplikacija konča z izračunavanjem podobnosti, nam na začetni zaslon prikaže spektrogram ter ime novo shranjene datoteke, pod tem pa še spektrogram in ime datoteke ki je najbolj podobna zajetemu zvoku (slika 5.3). Prav tako pri vsaki datoteki prikaže gumb za predvajanje posnetka, da se lahko prepričamo o pravilnosti razpoznave.

5.5 Shranjevanje parametriziranih vrednosti v predpomnilnik

Aplikacija v mapo AndroidRecorder zapisuje zgolj .wav datoteke. Kar bi pomenilo, da bi za vsak na novo zajet zvok aplikacija morala ponoviti prebiranje datoteke, parametrizacijo in izračun podobnosti. S tem bi zelo podaljšali čas razpoznavanja, če bi bilo shranjenih .wav datotek veliko. Zaradi hitrosti in odzivnosti aplikacije je bila razvita funkcionalnost predpomnilnika, ki omogoča, da ob zagonu aplikacije preberemo predpomnilniško datoteko in s tem pohitrimo delovanje. Predpomnilniška datoteka vsebuje serializirane objekte razreda Template. Objekt razreda Template tako shrani ime datoteke ter že parametrizirane vhodne podatke in jih shrani v datoteko cache.srl. V predpomnilniški datoteki so tako shranjeni podatki o datoteki, ki omogočajo, da izpustimo nepotrebno ponovno branje in parametrizacijo datotek.



Slika 5.3: Prikaz rezultatov razpoznavanja.

5.5.1 Brisanje podatkov

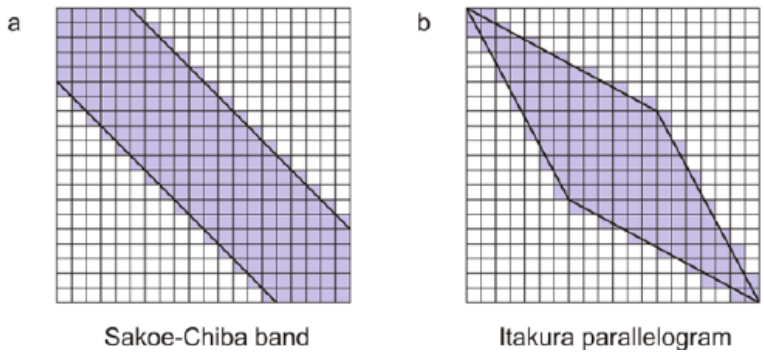
Ker aplikacija ne uporablja internega pomnilnika, ki je vezan na aplikacijo, smo pri aplikaciji ponudili tudi možnost brisanja shranjenih datotek in predpomnilnika. Če želimo podatke izbrisati preko orodja raziskovalec pa je dovolj, če pobrišemo vsebino mape AndroidRecorder.

5.6 Optimizacija delovanja razpoznavalnika

Pri delovanju razpoznavalnika uporabljamo princip dinamičnega programiranja in s tem povezane matrike podobnosti, zato lahko sklepamo, da je tukaj kar nekaj prostora za optimizacijo. Tako so v tem poglavju predstavljeni metodi zmanjševanja računske zahtevnosti izračuna matrike podobnosti in optimizacija izrisa rezultatov.

5.6.1 Zmanjševanje matrike podobnosti

V osnovnem delovanju razpoznavalnika, se matrika podobnosti izračuna za dva vhodna signala. Nato z dinamičnim programiranjem preverimo celotno matriko podobnosti in pridobimo optimalno ceno poti. Tako lahko z zmanjševanjem velikosti matrike, ki jo z dinamičnim programiranjem preverjamo, zmanjšamo čas potreben za izračun optimalne poti. Pri razpoznavanju govora se predvsem uporabljata Sakoe-Chiba pas in Itakura paralelogram, oba principa slonita na zmanjševanju uporabne velikosti matrike podobnosti.



Slika 5.4: Prikaz optimizacije matrike podobnosti z uporabo Sakoe-chiba pasu ter Itakura paralelograma.

Sakoe-Chiba pas predstavlja diagonalni pas iz začetne točke matrike M(0,0) do končne točke M(m,n) z v naprej definirano širino. S tem znatno zmanjšamo površino, ki jo z dinamičnim programiranjem preiskujemo. Lahko pa zaradi

enotne oblike pasu izgubimo optimalno pot, saj pas ne dopušča velikih odstopanj pri sredini matrike podobnosti. Tako je med bolj priljubljenimi načini omejevanje preiskovalnega prostora Itakura paralelogram, ki je kot si iz imena lahko predstavljamo, sestavljen iz paralelograma, ki poteka iz začetne točke matrike M(0,0) do končne točke M(m,n). Zaradi svoje oblike dopušča večja odstopanja pri sredini matrike podobnosti in s tem predstavlja bolj robustno rešitev napram Sakoe-Chiba pasu (slika 5.4).

Zaradi lažje implementacije se okviru optimizacije razpoznavalnika uporablja Sakoe-Chiba pas. Optimalna širina pasu za računanje podobnosti se izračuna kot absolutna razlika med dolžino in širino matrike podobnosti.

Z uporabo Sakoe-Chiba pasu je aplikacija ogromno pridobila na hitrosti, razpoznavanje pa je še vedno več kot zadovoljivo in v nekaterih primerih boljše kot brez optimizacije. Analiza delovanja standardne matrike podobnosti napram matriki podobnosti zmanjšane z Sakoe-Chiba pasu je predstavljena v sklopu analize razpoznavanja.

5.6.2 Izklop izrisa spektrograma

Pri uporabi predpomnilnika je bilo potrebno paziti na samo velikost, saj lahko z preveliko predpomnilniško datoteko izgubimo na hitrosti delovanja. Tako se v predpomnilnik ne shranjujejo podatki o spektrogramu, kar pomeni, da po končanem razpoznavanju ponovno izračunamo spektrogram iz dveh razpoznanih signalov. S tem nekoliko izgubimo na hitrosti delovanja, kar pa nadoknadimo s hitrostjo branja majhne predpomnilniške datoteke. Tako je bila kot drugi del optimizacije delovanja razpoznavalnika razvita možnost vklopa in izklopa izrisa spektrogramov po opravljenem razpoznavanju. Kar je v povprečju prineslo k 200-500ms hitrejšemu delovanju razpoznavalnika. Če v obzir vzamemo velikost testnih primerov to pomeni do 60% hitrejše delovanje. Še vedno pa se ob izklopljenem izrisu spektrograma pri rezultatu pojavijo gumbi za predvajanje razpoznanih signalov.

5.7 Analiza vpliva optimizacije

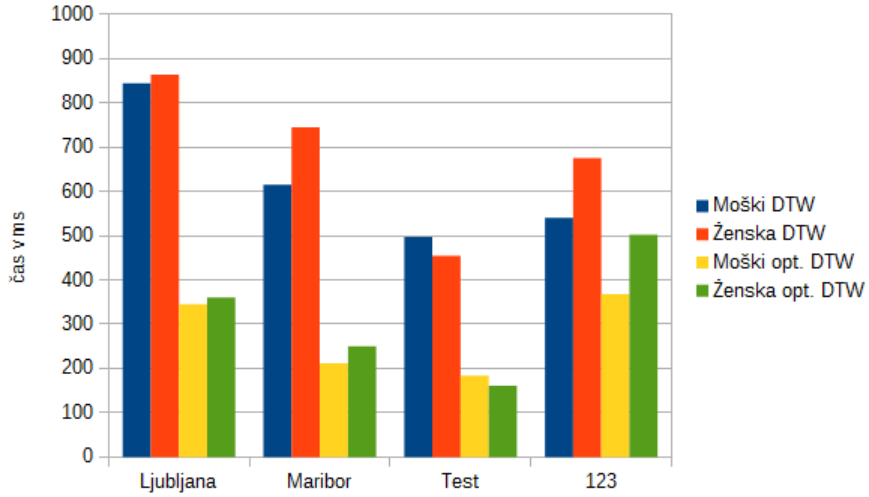
V okviru diplomske naloge je bila opravljena tudi analiza delovanja razpoznavalnika. Tako je v prvem podpoglavju predstavljena primerjava hitrosti pred in po optimizaciji razpoznavalnika. V drugem podpoglavju se nahaja analiza razpoznavanja na večji testni množici.
Pri analizi je bil uporabljen generični razred TimingLogger, ki se uporablja za beleženje časa porabljenega pri izvajanju različnih funkcij. V primeru naše aplikacije smo merili čas izvajanja algoritma časovnega izkrivljanja ter izris spektrogramov. Z uporabo funkcije addSplit(»tekst«) se v objekt razreda TimingLogger doda nov vmesni čas izvajanja. Ko smo zaključili s spremljanjem pa z uporabo funkcije dumpToLog() v dnevnik AndroidStudia izpišemo rezultate. Tako pridobljeni rezultati lahko še vedno minimalno varirajo glede na trenutne procese v napravi.

Za zagotavljanje robustnosti razpoznavalnika, sta pri analizi sodelovali dve osebi (moški in ženska). Tako spodaj predstavljeni rezultati predstavljajo dejansko uporabo in natančnost razpoznavalnika.

5.7.1 Primerjava hitrosti razpoznavanja

Aplikacija z uporabo Sakoe-Chiba pasu za zmanjševanje matrike podobnosti znatno pridobi na hitrosti. Tako ta del analize prikazuje razliko v hitrosti preračunavanja algoritma časovnega izkrivljanja pred in po optimizaciji. Pri rezultatih analize se uporabljajo časi pravilno razpoznanih besed.

Pri izvajanju analize sta testna govorca v večni primerov uporabljala isto hitrost izgovorjave. V zadnjem primeru pa uporabljata različno hitrost izgovorjave pred in po optimizaciji, kar opazimo pri času izračuna optimalne poti. Kljub veliki razliki v hitrosti izgovorjave je lahko opaziti pohitritev izračuna optimalne poti optimiziranega algoritma.



Slika 5.5: Prikaz hitrosti delovanja razpoznavalnika pred in po optimizaciji.

optimizacija/zvok	`Ljubljana`	`Maribor`	`Test`	`123`
`brez`	$844ms$	$615ms$	$497ms$	$540ms$
`optimiziran DTW`	$345ms$	$212ms$	$184ms$	$368ms$

Tabela 5.1: Analiza optimizacije z uporabo moškega glasu

optimizacija/zvok	`Ljubljana`	`Maribor`	`Test`	`123`
`brez`	$864ms$	$745ms$	$455ms$	$675ms$
`optimiziran DTW`	$360ms$	$250ms$	$161ms$	$502ms$

Tabela 5.2: Analiza optimizacije z uporabo ženskega glasu

Z uporabo optimiziranega algoritma časovnega izkrivljanja tako v povprečju zmanjšamo čas izračuna optimalne poti za 50%.

5.7.2 Primerjava uspešnosti razpoznavanja

Pri uporabi optimizacije je ob povečanju hitrosti potrebno primerljivo ohraniti tudi uspešnost razpoznavanja. Ta del analize zato vsebuje pregled točnosti razpoznavalnika ob zmanjševanju matrike podobnosti do optimalne širine Sakoe-Chiba pasu.

Testna množica je sestavljena iz 5 zvočnih primerov. Za izračun povprečja pa se naredi 10 razpoznavanj za vsako stopnjo optimizacije.
Analizo začnemo z neoptimizirano matriko podobnosti nato pa zmanjšujemo širino Sakoe-Chiba pasu do optimalne širine.

Širina pasu 100% prestavlja širino pasu definirano kot absolutno razdaljo med dolžino in širino izračunane matrike podobnosti.

stopnja optimizacije	št. pravilno razpoznanih zvokov	uspešnost razpoznavanja
brez optimizacije	6	60%
širina pasu 200%	7	70%
širina pasu 150%	7	70%
širina pasu 125%	7	70%
širina pasu 100%	8	80%
širina pasu 90%	1	10%

Tabela 5.3: Analiza optimizacije z zmanjševanjem matrike podobnosti.

Pri izvedbi analize sta bila uporabljena tako moški kot ženski glas, čemur tudi pripisujemo manjšo uspešnost razpoznavanja. Ob zmanjševanju širine pasu do optimalne meje (v našem primeru 100%) se uspešnost razpoznavanja celo

izboljša. Če se spustimo pod optimalno mejo 100%, uspešnost razpoznavanja bistveno pade (tabela 5.3).

Poglavje 6

Sklepne ugotovitve

V sklopu diplomskega dela je bila razvita aplikacija za razpoznavanje govora, ki svojo funkcijo opravi brez uporabe spletne povezave in večjega števila učnih primerov. Ogrodje za zajem zvoka je na platformi Android dobro definirano in podprto z dokumentacijo. Večji del težav pa so predstavljale vhodno izhodne zmogljivosti formata .wav, ki na Android platformi ne dobi več podpore kot le predvajaj in shrani datoteko. Tako je bila pri diplomskem delu uporabljena in dopolnjena odprto kodna knjižnica za delo z .wav formatom WaveTools.

Izdelana aplikacija ima z morebitnim nadaljnjim razvojem zanimivo prihodnost za uporabo tudi v učne namene. Z trenutno modularnostjo aplikacije lahko z majhnimi popravki dodamo veliko funkcionalnosti, ki doprinesejo k lažjemu razumevanju postopkov razpoznavanja govora.. Večjo praktično uporabo pa ob veliki popularnosti privzetih razpoznavalnikov na platformah iOS in Android ni pričakovati.

6.1 Možne nadgradnje obstoječe aplikacije

Pri razvoju aplikacije smo dobili veliko dodatnih zanimivih idej. Tako bi lahko pri uporabi algoritma časovnega razpoznavanja implementirali še kakšno

drugo vrsto optimizacije poleg trenutne. Prav tako kar nekaj prostora za nadgradnjo predstavlja prikaz razpoznavanja. Vsekakor pa bi lahko poleg trenutne metode razpoznavanja implementirali še kakšno in tako omogočili lažjo primerjavo med različnimi metodami razpoznavanja.

6.2 Spremna misel

Razvoj aplikacije je bil v prvi vrsti zanimiv in poučen. Ob tem smo pridobili ogromno novega znanja ter boljši vpogled v načrtovanje projektov in razporejanje časa. Kljub trenutno manjši popularnosti razpoznavalnikov zvoka pa menimo, da je prihodnost svetla in upamo, da z razvojem aplikacije pripomoremo k lažjemu razumevanju tovrstnih tehnologij.

Literatura

[1] GNU general public license. Dosegljivo: https://github.com/f02h/androidASR/blob/master/LICENSE [Dostopano 14.3.2017]

[2] Bill Phillips, Chris Stewart, Brian Hardy in Kristin Marsicano, "Android Programming: The Big Nerd Ranch Guide (2nd Edition)",Big Nerd Ranch,2015

[3] Software Freedom Conservancy, "Documentation" Dosegljivo: https://git-scm.com/doc [Dostopano 1.11.2016].

[4] Jeff Sutherland,J.J. Sutherland, "Scrum: The Art of Doing Twice the Work in Half the Time", Crown Business, 2014

[5] Jont B. Allen, "Short Time Spectral Analysis, Synthesis, and Modification by Discrete Fourier Transform" ,v zborniku: IEEE Transactions on Acoustics, Speech, and Signal Processing, 1977.

[6] R. Rozman ,"Enostavnejša zasnova sistema za razpoznavanje govora",Elektrotehniški vestnik Letn. 80, št. 4 ,2013, str. 171-176.

[7] Wikipedia, "DTW - wikipedia the free encyclopedia" Dosegljivo:https://en.wikipedia.org/wiki/Dynamic_time_warping [Dostopano 5.11.2016].

[8] Bhadragiri Jagan Mohan, Ramesh Babu. N, "Speech Recognition using MFCC and DTW" ,v zborniku: 1st Int. Conf. on Advances in Electrical Engineering, Vellore, India, jan. 2014.

[9] S.B. Davis, and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentence" ,v zborniku: IEEE Transactions on Acoustics, Speech, and Signal Processing, 1980.

[10] R.K. Sharma, Tripti Kapoor, "Application of Automatic Speaker Recognition techniques to Parkinsons's disease diagnosis" ,v zborniku: 2nd International Conference on Signals, Systems & Automation (ICSSA 2011) & 1st International Conference on Intelligent Systems & Data Processing, 2011, str. 127-130.

[11] Wikipedia, "Speech recognition - wikipedia the free encyclopedia" Dosegljivo: https://en.wikipedia.org/wiki/Speech_recognition [Dostopano 5.11.2016]

[12] Columbia University, "Filterbank Analysis" Dosegljivo: http://www.ee.columbia.edu/ln/rosa/doc/HTKBook21/node54.html [Dostopano 5.11.2016]